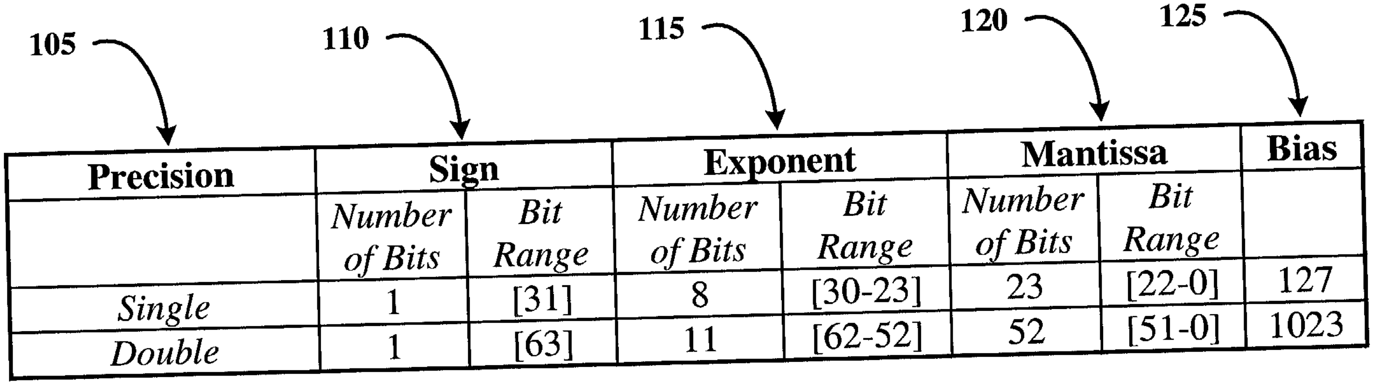


Precision	**Sign**		**Exponent**		**Mantissa**		**Bias**
	Number of Bits	*Bit Range*	*Number of Bits*	*Bit Range*	*Number of Bits*	*Bit Range*	
Single	1	[31]	8	[30-23]	23	[22-0]	127
Double	1	[63]	11	[62-52]	52	[51-0]	1023

Figure 1A

Floating Point Numbers
Under IEEE Standard 754

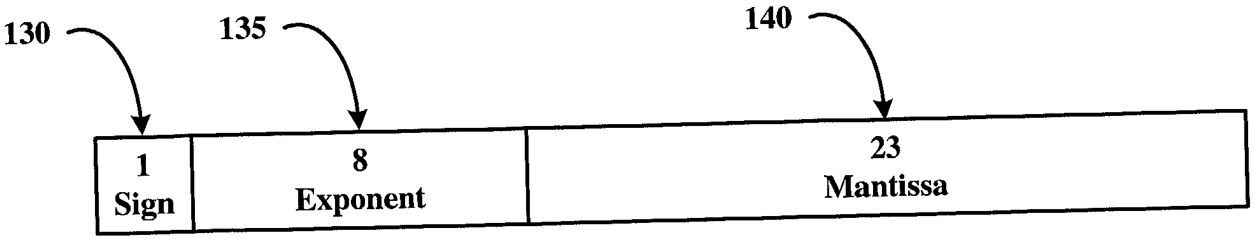


Figure 1B

Floating Point
Single Precision Number

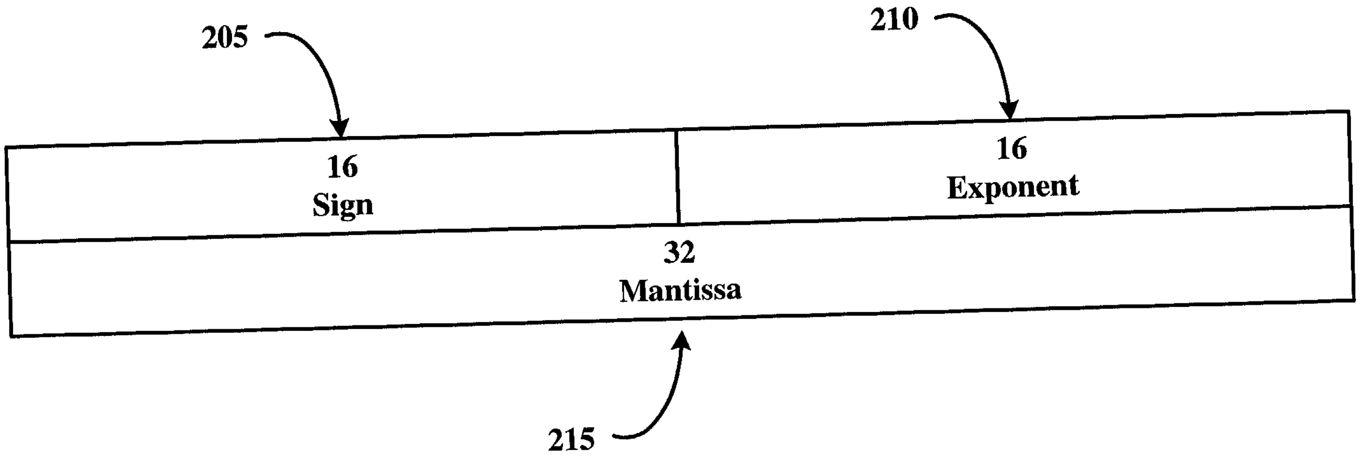
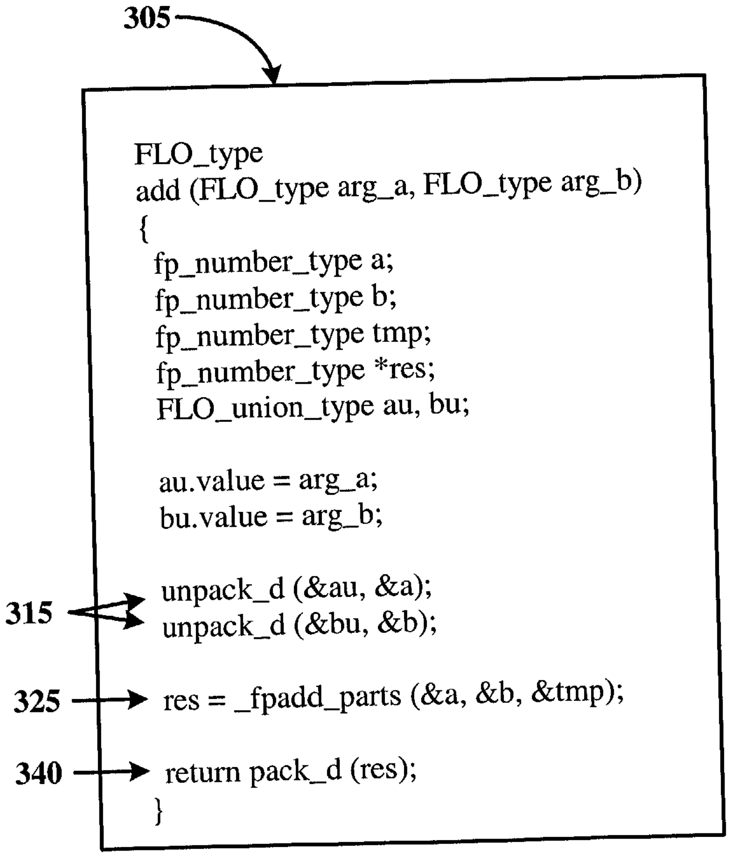


Figure 2

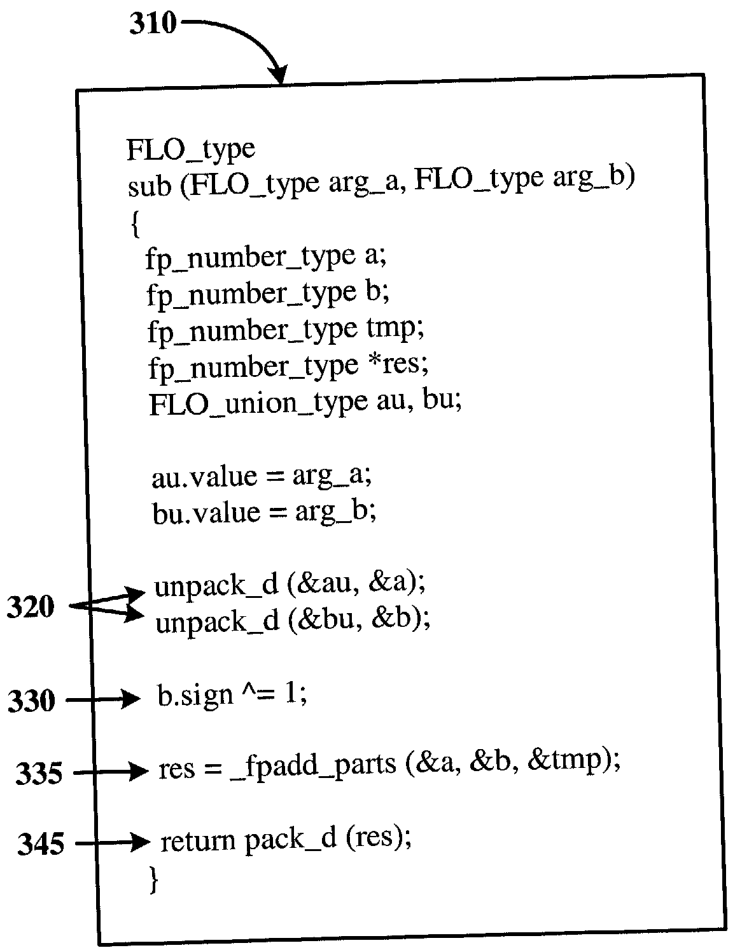
Unpacked
Floating Point Number



```
FLO_type
add (FLO_type arg_a, FLO_type arg_b)
{
 fp_number_type a;
 fp_number_type b;
 fp_number_type tmp;
 fp_number_type *res;
 FLO_union_type au, bu;

 au.value = arg_a;
 bu.value = arg_b;

 unpack_d (&au, &a);
 unpack_d (&bu, &b);

 res = _fpadd_parts (&a, &b, &tmp);

 return pack_d (res);
}
```

Figure 3A

Conventional Addition Routine



```
FLO_type
sub (FLO_type arg_a, FLO_type arg_b)
{
 fp_number_type a;
 fp_number_type b;
 fp_number_type tmp;
 fp_number_type *res;
 FLO_union_type au, bu;

 au.value = arg_a;
 bu.value = arg_b;

 unpack_d (&au, &a);
 unpack_d (&bu, &b);

 b.sign ^= 1;

 res = _fpadd_parts (&a, &b, &tmp);

 return pack_d (res);
}
```

Figure 3B

Conventional Subtraction Routine

405 $T = a \times b \times c$

410 $T_0 = a \times b$

415 $T_1 = T_0 \times c$

420 $T_2 = unpack(a)$

425 $T_3 = unpack(b)$

430 $T_4 = unpack\_mult(T_2, T_3)$

435 $T_0 = pack(T_4)$

440 $T_5 = unpack(T_0)$

445 $T_6 = unpack(c)$

450 $T_7 = unpack\_mult(T_5, T_6)$

455 $T_1 = pack(T_7)$

Figure 4

Calculation Using Conventional Floating Point Emulation

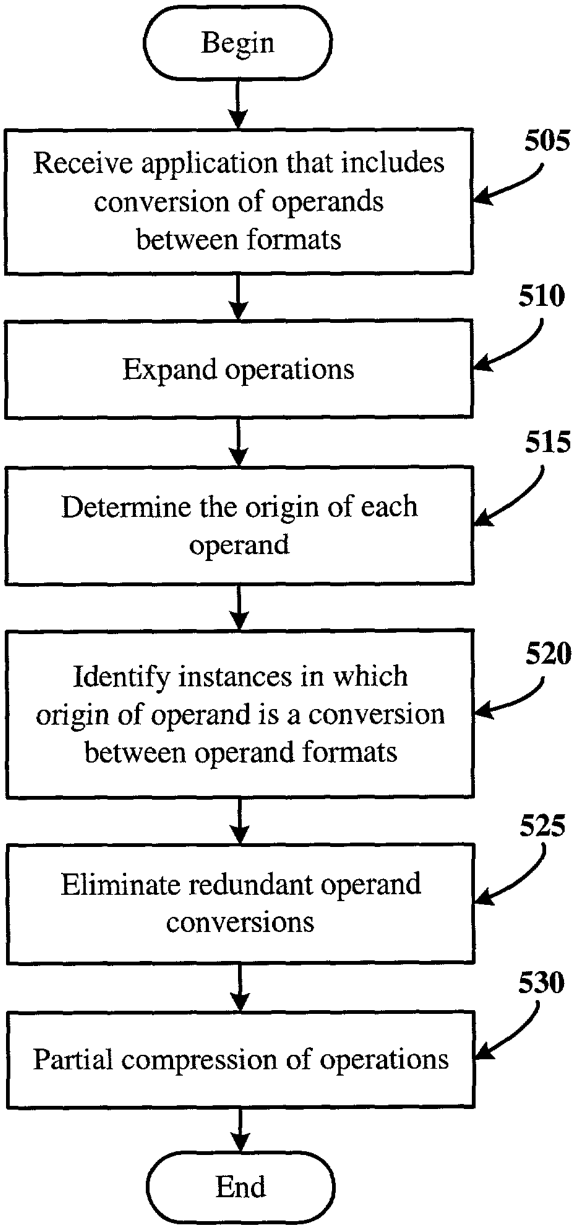


Figure 5

Operand Conversion Optimization

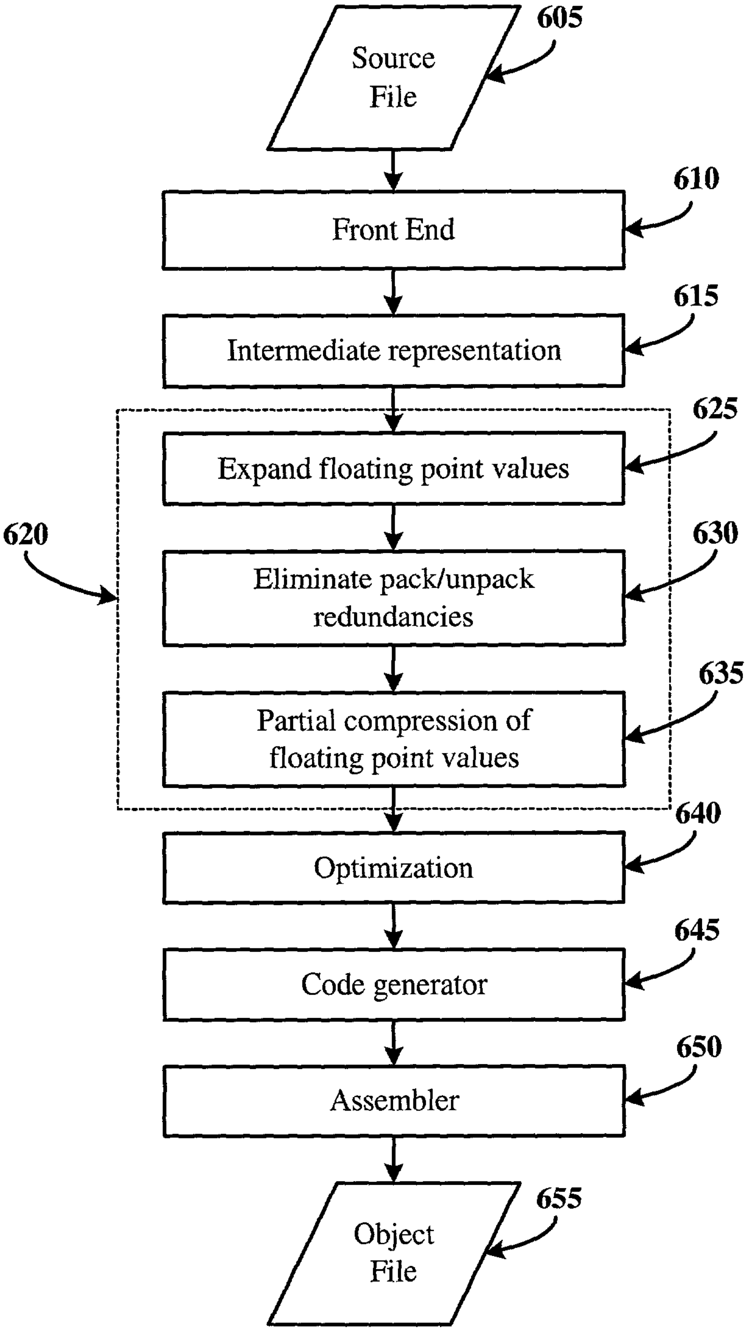


Figure 6

Exemplary System Including Operand Conversion Optimization

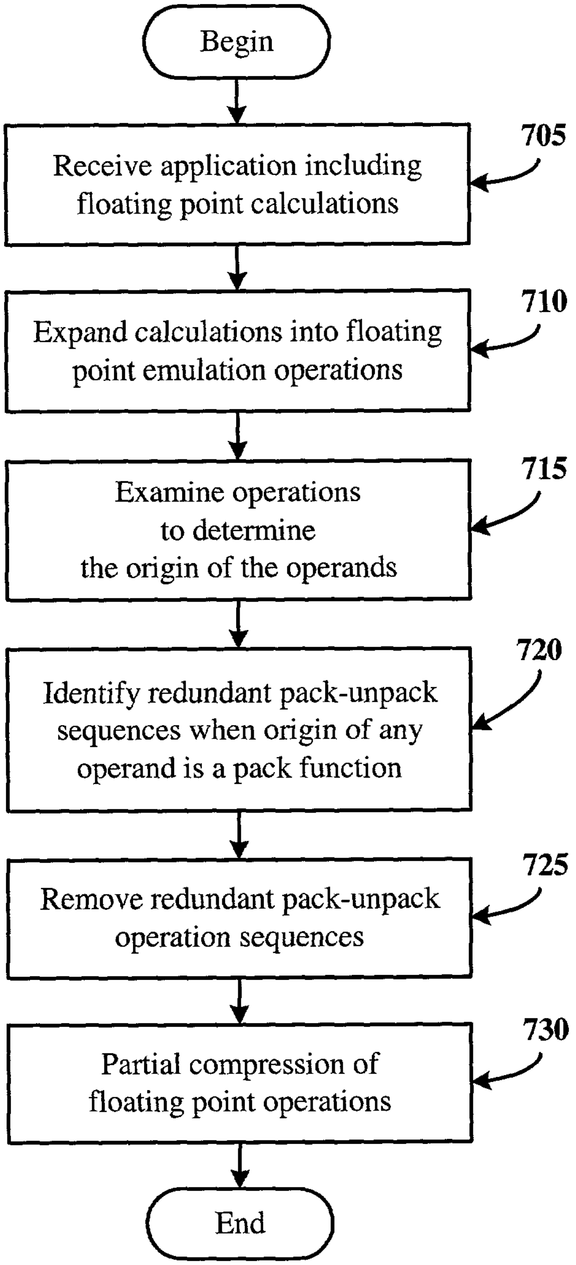


Figure 7

Optimization of Floating Point Emulation

805 $T = a \times b \times c$

810 $T_0 = a \times b$

815 $T_1 = T_0 \times c$

820 $T_2 = unpack(a)$

825 $T_3 = unpack(b)$

830 $T_4 = unpack\_mult(T_2, T_3)$

835 $T_5 = unpack(c)$

840 $T_6 = unpack\_mult(T_4, T_5)$

845 $T_1 = pack(T_6)$

Figure 8

Calculation Using
Emulation Optimization

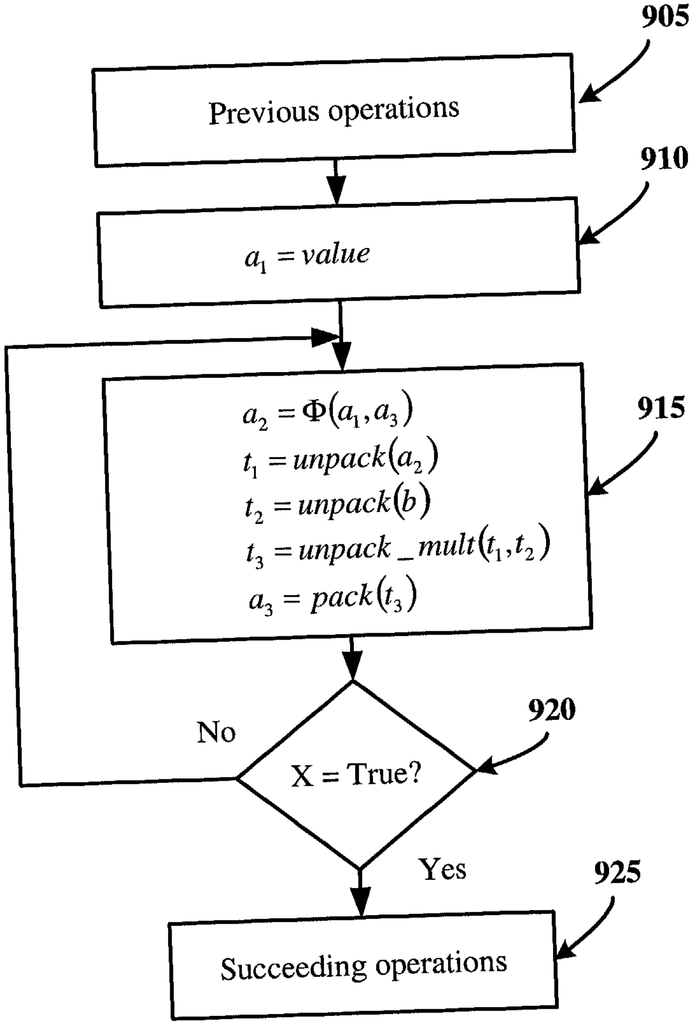


Figure 9A

Loop with
Partial Redundancy

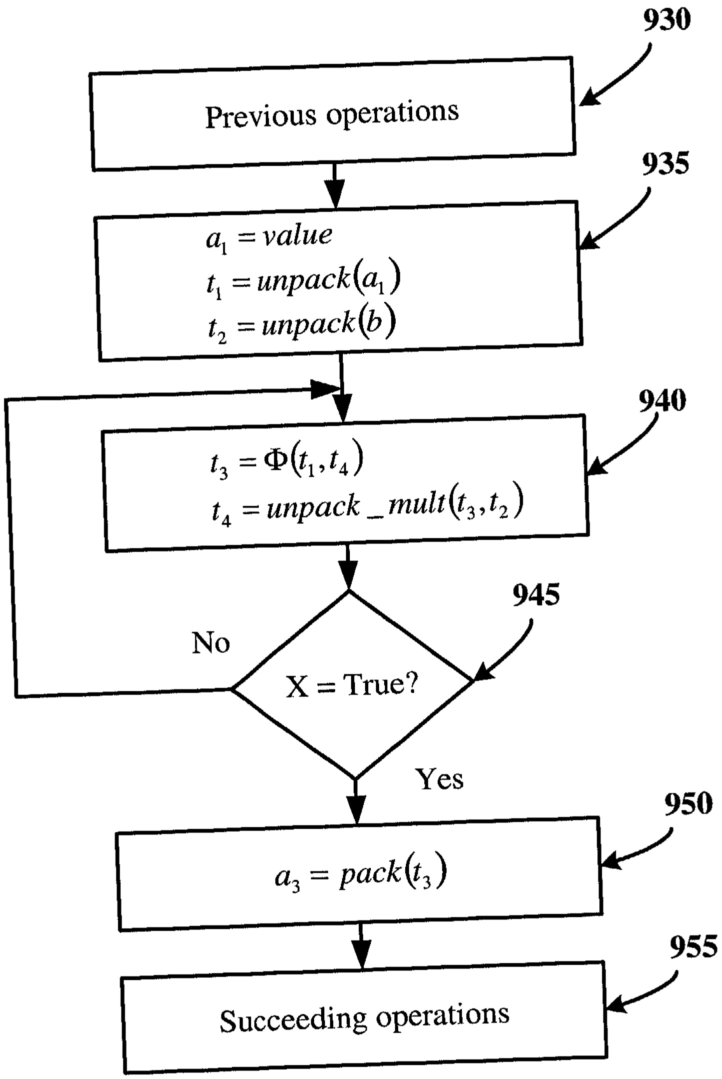


Figure 9B

Optimized
Loop